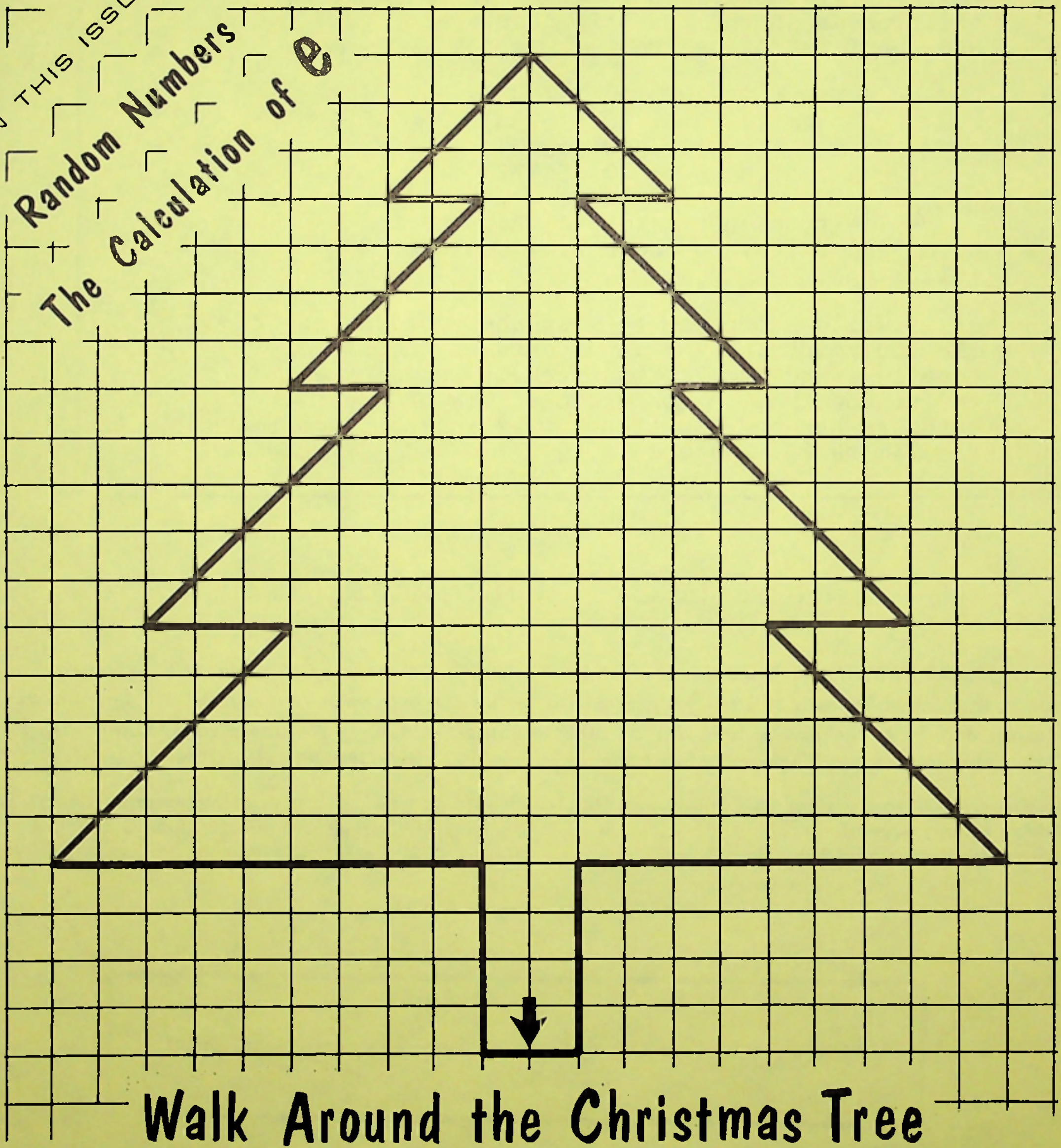
# Popular Computing

21

DECEMBER

74

VOL 2 NO 12



Walk Around the Christmas Tree

# The Christmas Tree Walk

PROBLEM 69

Start at the top of the tree and travel clockwise around the tree, taking steps of .3 units. If the origin is taken at the base of the tree as shown, the trip starts at (0,21) and the first steps are:

```
(.212132, 20.787686)
(.424264, 20.575736)
(.636396, 20.363604)
(.848528, 20.151472)
```

The total perimeter of the tree is 90.083259 units, so there will be just 300 full steps.

Write a program to list the coordinates of the 300 points. The input data is the set of equations and the limits on their ranges that define the tree. For example, one of the branches has the equation y = -x + 17 between the limits (3, 14) and (8, 9).

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

**POPULAR COMPUTING** is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $18 per year, or $15 if remittance accompanies the order. For Canada and Mexico, add $4 to the above rates. For all other countries, add $6 to the above rates. Back issues $2 each. Subscriptions may begin with any issue. Copyright 1974 by POPULAR COMPUTING.

Publisher: Fred Gruenberger
Editor: Audrey Gruenberger
Associate editor: David Babcock
Irwin Greenwald
Contributing editors: Richard Andree
Daniel D. McCracken
William C. McGee
Advertising manager: Ken W. Sims
Art director: John G. Scott

The essence of randomness is unpredictability. For various interesting computing situations (sampling, shuffling, permuting, or solving problems by the technique known as Monte Carlo), we seek numbers such that, in an essentially unending stream, no matter how much information is accumulated about the stream, there is no clue as to what will come next. Further, we want this unpredictable information to be controlled; that is, to be generated in some way that can be precisely repeated.

This task is impossible; in fact, the two chief characteristics are inconsistent with each other. If the stream of generated numbers is unpredictable, then it can't be repeated, and vice versa. Notice the term "generated." It is of little practical use to have access to a table of random numbers (or random digits). Whatever use may be made of the stream of unpredictable numbers, we usually need a lot of them, perhaps billions of them. A table that will fill all of core storage of a large machine will be exhausted in a few seconds in most Monte Carlo work. We seek schemes to generate the numbers or digits by an algorithm, and in so doing we will have perfectly predictable numbers and hence, by definition, not random numbers.

It is a dilemma, and it leads to what is perhaps the only piece of solid illogic in all of science. We say "if there were a stream of truly random digits, what would be their characteristics? These numbers (that we can generate) have the same characteristics, and therefore they can be considered random." Since that line of reasoning is a non sequitur, we fudge by calling our generated numbers "pseudo-random."

Anyone wishing to know the theory of random number generation or the details of the standard tests to be applied to generated numbers should consult the references at the end of this chapter, and particularly Knuth.

* * *

Knuth will tell you more about the subject than you can absorb in a year; Kendall and Babington-Smith were the pioneers, and provided the first tests of randomness; Hull and Dobell's article, although now quite old, is thorough and includes an extensive bibliography. The following quotations from Knuth are noteworthy:

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

--John von Neumann

" The most prudent policy for a person to follow is to run each Monte Carlo program at least twice using quite different sources of random numbers, before taking the answers of the program seriously; this not only will give an indication of the stability of the results, it also guards against the chance of trusting in a generator with hidden deficiencies. (Every random-number generator will fail in at least one application.)"

"Many random-number generators in use today are not very good. There is a tendency for people to avoid learning anything about random-number generators; quite often we find that some old method which is comparatively unsatisfactory has blindly been passed down from one programmer to another, and today's users have no understanding of its limitations."

"...random numbers should not be generated with a method chosen at random. Some theory should be used."

"The reader probably wonders, 'Why are there so many tests?' It has been said that more computer time has been spent testing random numbers than using them in applications!"

"Perhaps the main reason for doing extensive testing on random-number generators is that people misusing Mr. X's random-number generator will hardly ever admit that their programs are at fault: they will blame the random-number generator, until Mr. X can _prove_ to them that his numbers are sufficiently random. On the other hand, if the source of random numbers is only for Mr. X's personal use, he might decide not to bother to test them..."

"...a truly random sequence will exhibit local nonrandomness; local nonrandomness is necessary in some applications, but it is disastrous in others. We are forced to conclude that _no sequence of 'random' numbers can be adequate for every application_."

The person using a random number generator should know something about the theory. Ideally, he should construct his own generator and test it thoroughly, and in particular should test it specifically for the use to which it is to be put. Most people, however, do not feel that they can afford that much effort, and simply want a generator that someone has certified. Knuth (pages 155-156) provides a set of guidelines for the purpose of constructing a generator that will not embarrass its user.

_The_ standard generator is this:

$$x_{n+1} = (M \cdot x_n + C) \bmod P$$

which, for suitable choices of M, P, and C, provides a recursion that will yield a string of numbers, given a suitable (non-zero) starting value, $x_0$. For purposes of exposition, let us use C = 0, M = 2, and P = 19, with $x_0 = 1$. The recursion then produces this string of numbers: 2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1 and then repeats. The cycle length is 18 numbers; we get all the numbers from 1 to 18 (that is, from 1 to P-1) in somewhat scrambled order. The scrambling effect is not too good. Notice that small numbers stay small for quite a while (1, 2, 4, 8...) and large numbers (i.e., those near P) stay large.

Let's repeat this exercise, again using C = 0, but with M = 3, and P = 29. We now get the sequence 3, 9, 27, 23, 11, 4, 12, 7, 21, 5, 15, 16, 19, 28, 26, 20, 2, 6, 18, 25, 17, 22, 8, 24, 14, 13, 10, 1. Again, we get the maximum cycle length (that is, all 28 numbers between 1 and P-1) and the scrambling effect is somewhat better.

By increasing the modulus, P, we can make the cycle length as long as we please. For P = 40001387, for example, the cycle length is 40001386 (provided that the multiplier, M, is relatively prime to P). The generator

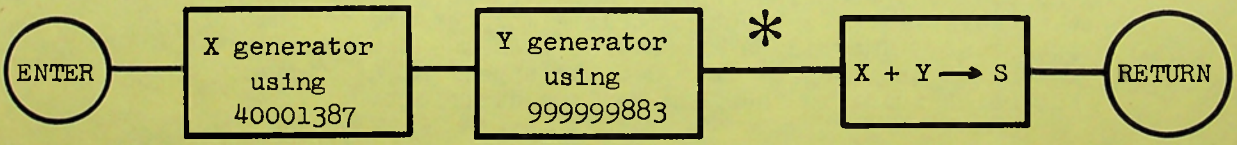
$$x_{n+1} = 13^5 x_n \bmod 40001387$$

will produce the 40001386 numbers between 1 and 40001386 in fair scrambled order. By itself, this could be called a random number generator, but one should be careful. Notice, for example, that the high order digit of the output stream is almost always 0, 1, 2, or 3, which makes the output rather predictable.

Consider now a similar generator:

$$y_{n+1} = 5^{13} y_n \bmod 999999883$$

which has a cycle length of nearly a billion numbers. If we used both of the above generators in a subroutine like this:



the output stream of S values has a cycle length of around $10^{16}$ numbers before repeating (although individual numbers will repeat, of course); the local effects of the separate generators are suppressed; if the initial values for x and y are reset to their original (non-zero) values, the output stream will repeat precisely; the output stream will pass most of the tests for randomness; and the resulting generator is fairly idiot-proof, in that there will be few side effects (e.g., such things as having all the output numbers divisible by 17). In the event that this generator is unsatisfactory for any reason, a third generator:

$$z_{n+1} = 7^5 z_n \bmod 999999893$$

can be added (at the place marked * on the flowchart, and with x, y, and z added into S) to further homogenize the output and, incidentally, to drive the cycle length up to around 10 to the 24th power.

For most computers, the particular moduli used in the separate generators given above are too long to be used easily. The notion of using simple generators and "beating" several of them against each other in one subroutine leads to a generator that has certain virtues. Such a generator, written in Fortran, is given at the end of this article. The moduli of the three independent sub-generators are 98801, 98899, and 99989 (these being primes just under 100000 having a primitive root of 2). The multipliers used are all 2, and the sum of the outputs of the sub-generators is reduced modulo 100003. The subroutine requires that the Fortran compiler be able to handle integers up to 100003. If this is not the case, the following moduli can be substituted:

32603 32507 32183

Knuth's advice is excellent, especially the admonition marked with ✱. I feel that it is even more important that the user of any generator (for that matter, the user of any packaged program) perform his own tests and in particular, that he test the generator for the specific characteristics that he wants it to exhibit. For example, suppose one wishes to perform a simulation of a scheme for winning at dice. The scheme is programmed, and that includes a subroutine for the game of dice, and that in turn calls for a subroutine that simulates the tossing of two dice. That is, a subroutine is needed whose output is one of the numbers from 2 to 12 appearing at random but maintaining in the long run the distribution:

| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1/36 | 1/18 | 1/12 | 1/9 | 5/36 | 1/6 | 5/36 | 1/9 | 1/12 | 1/18 | 1/36 |

Thus, a run of 36000 calls of that subroutine should yield a distribution close to (but not too close to):

| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 5000 | 4000 | 3000 | 2000 | 1000 |

and the chi-squared test for goodness of fit will settle the "close to but not too close" question. That may not be sufficient. If the theory being tested rests heavily on the sequence of dice tosses (for example, that three 4's in a row should occur approximately ten times in 17000 tosses of two dice), then those characteristics should be tested for. There seems little point to insisting that the generator being used pass the Coupon Collector's Test (which is one of the 8 standard tests of randomness), if that particular characteristic is irrelevant to the planned use. On the other hand, if the generator is to be used to determine the coordinates of points in the plane, then it should be able to pass the $d^2$ test (see Reference 5) and it would be well to subject it to that test.

Most of the research on random number generators (next to sorting, random number generators have been the leading topic in computing literature for the past 20 years) has been concerned with generators of high efficiency in the sense of producing acceptable output with a minimum use of CPU time or storage space. These goals are commendable, but I much prefer to use generators that have extremely long cycle lengths and that are extremely stable, even though these two characteristics may lead to waste of CPU time and/or storage space. The generator given at the end of this article is recommended--but only if the user subjects it to some tests of his own.

REFERENCES

1. Knuth, Donald E., The Art of Computer Programming, Vol. 2, Chapter 3, "Random Numbers," Addison-Wesley, 1972.

2. Hull, T. E., and A. R. Dobell, "Random Number Generators," SIAM Review, Vol. 4, 1962.

3. Kendall, M. G., and B. Babington-Smith, "Randomness and Random Sampling Numbers," J. Roy. Stat. Soc., Vol. 101, 1938.

4. Lehmer, D. H., "Mathematical Methods in Large-scale Computing Units," Proc. Second Symposium on Large-Scale Digital Calculating Machinery, 1949, Harvard University Press.

5. Gruenberger, F., "The $d^2$ Test of Random Digits," Mathematical Tables and Other Aids to Computation, Vol. 5, 1951.

6. Metropolis, N. C., G. Reitwiesner, and J. von Neumann, "Statistical Treatment of Values of First 2000 Decimal Digits of e and of pi Calculated on the ENIAC," Mathematical Tables and Other Aids to Computation, Vol. 4, 1950.

```
      CØMMØN IR(3)
C      FØUR NUMBERS (FRØM 1 TØ 100000 EACH) MUST BE ENTERED
C      BEFØRE THE GENERATØR IS CALLED THE FIRST TIME.
C      AFTER THAT ØNLY THE LAST RANDØM NUMBER (NUMBER)
C      NEED BE PASSED.
      READ 2, NUMBER, IR
    1 CALL RANDM(NUMBER)
      PRINT 2,NUMBER
    2 FØRMAT(I6)
      GØ TØ 1
      END
      SUBRØUTINE RANDM (N)
      CØMMØN I, J, K
      I=MØD(I*2,98801)
      J=MØD(J*2,98899)
      K=MØD(K*2,99989)
      N=N+I+J+K
      N=MØD(N,100003)
      RETURN
      END
```

The subroutine $\boxed{\text{in Fortran}}$ begins at: SUBRØUTINE RANDM (N)

Initialize, in housekeeping, by assigning any non-zero, positive integers to:

IR(1)
IR(2)
IR(3)
NUMBER
} each less than 100000

The output from the subroutine, NUMBER, will be an integer:

$0 \leq \text{NUMBER} \leq 100002$

(Lee Armer, April, 1971)

END

# The Calculation of e

Inasmuch as the number e, the natural logarithm base, has been calculated to 105,000 decimal places, there is little need to calculate it again. However, in line with our belief that

THE WAY TO LEARN COMPUTING IS TO COMPUTE

the recalculation of this ubiquitous constant, to a few hundred decimal places, provides an interesting and challenging problem, particularly since the number e is one of the simplest transcendental numbers to compute. The calculation shown below represents about 15 minutes of work with a pocket calculator, and yields 20 significant digits of e (to be sure, the calculation of subsequent digits will require considerably more effort).

| | Quotient | Remainder | Quotient | Remainder | Quotient | Remainder |
|---|---|---|---|---|---|---|
| 0 | 1.0000000 | | | | | |
| 1 | 1.0000000 | | | | | |
| 2 | .5000000 | 00 | 0000000 | | | |
| 3 | .1666666 | 02 | 6666666 | 02 | 6666666 | 02 |
| 4 | .0416666 | 02 | 6666666 | 02 | 6666666 | 02 |
| 5 | .0083333 | 01 | 3333333 | 01 | 3333333 | 01 |
| 6 | .0013888 | 05 | 8888888 | 05 | 8888888 | 05 |
| 7 | .0001984 | 00 | 1269841 | 01 | 2698412 | 04 |
| 8 | .0000248 | 00 | 0158730 | 01 | 1587301 | 04 |
| 9 | .0000027 | 05 | 5573192 | 02 | 2398589 | 00 |
| 10 | .0000002 | 07 | 7557319 | 02 | 2239858 | 09 |
| 11 | .0000000 | 02 | 2505210 | 09 | 8385441 | 07 |
| 12 | | 00 | 0208767 | 06 | 5698786 | 09 |
| 13 | | 00 | 0016059 | 00 | 0438368 | 02 |
| 14 | | 00 | 0001147 | 01 | 0745597 | 10 |
| 15 | | 00 | 0000076 | 07 | 4716373 | 02 |
| 16 | | 00 | 0000004 | 12 | 7794773 | 05 |
| 17 | | 00 | 0000000 | 04 | 2811457 | 04 |
| 18 | | | | 00 | 0156192 | 01 |
| 19 | | | | 00 | 0008220 | 12 |
| 20 | | | | 00 | 0000411 | 00 |
| 21 | | | | 00 | 0000019 | 12 |
| 22 | | | | 00 | 0000000 | 19 |
| | 2.7182814 | | 4\|2845898 | | 6\|5235350 | |
| | 4 | | 6 | | | |
| | 2.7182818 | | 2845904 | | 5235350 | |

The number e is the only number that can be directly raised to any power, which is why it forms the base for natural logarithms. The definition of the function $e^x$ implies that all its derivatives are equal to the function itself, which makes $e^x$ the ideal function for expansion in a Taylor series (and most calculus texts demonstrate the application of Taylor series by using $y = e^x$). In simplified form, the derivation goes like this. Assume that a function can be represented by an infinite power series:

$$f(x) = a + bx + cx^2 + dx^3 + ex^4 + fx^5 + \cdots$$
$$e^x = a + bx + cx^2 + dx^3 + ex^4 + fx^5 + \cdots$$

If this is permissable, then the equality should hold for $x = 0$, so we have:

$$e^0 = 1 = a.$$

Moreover, if the proper conditions are met, the derivatives of both sides of the equation should be equal:

$$\frac{d}{dx}(e^x) = e^x = b + 2cx + 3dx^2 + 4ex^3 + 5fx^4 + \cdots$$

and this new equation should also hold when $x = 0$, so that:

$$e^0 = 1 = b.$$

This procedure can be continued (i.e., differentiate both sides and set $x = 0$), and the successive coefficients are determined one at a time. We arrive at:

$$e^x = 1 + x + x^2/2 + x^3/6 + x^4/24 + x^5/120 + \cdots$$

and now, letting $x = 1$, we have a series whose sum is e itself:

$$e = 1 + 1 + 1/2 + 1/6 + 1/24 + 1/120 + \cdots$$

The calculation for the first 8 digits is shown in the first two columns of the Figure. Successive terms are found by dividing the preceding term by the term number. On each line, the remainder of the division is recorded, so that the process can continue for another 7 digits of quotient. The sums of the quotient columns each (after the first) have a carry to the preceding sum, as shown by the arrows. Each new series of quotients ends when a quotient of zero is reached.

The process has been illustrated with 7-digit quotients, but will work with quotients of any size. The most convenient size is the normal one-word quotient of your machine. For a modest calculation of e (say, a hundred digits or so), all the remainders can be kept in central storage. If the calculation is to be extended further, the remainders might have to be recorded in an external medium, such as magnetic tape. All the quotients were displayed in the illustration; in practice, they can be summed along the way, so that all that needs to be stored is the sums and the remainders.

The calculation of e is perhaps the simplest high precision calculation that can be made. The advent of computers has made it relatively easy to perform arithmetic to any desired degree of precision, but there has actually been very little of it done. If we take "high precision" to mean 500-digit or better, then the following is probably a fairly complete list of all that has been accomplished to date:

1. Pi to 1,000,000 digits.

2. e to 100,000 digits.

3. $\sqrt{2}$ to 1,000,000 digits; one or two other square roots to 1000 digits.

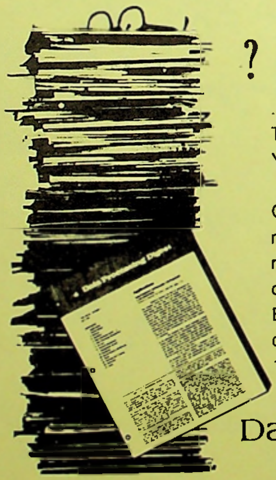
4. The real root of Wallis' equation ($x^3 -2x -5 = 0$) to 1000 digits.

5. Exact factorials, all values up to 1000! and scattered values to 10000!

6. The Fibonacci sequence, to the 1000th term.

7. A few scattered numbers that have appeared in POPULAR COMPUTING, such as the cube root of 16 (PC16-18), the sine of one radian (PC12-1) and some 1000th powers in the N-series.

8. And, of course, that endless search for the next "largest known prime number," for which the current title holder is the 6002-digit number shown in PC19-6.



? FRUSTRATED

. . . SEARCHING FOR INFORMATION?
TIME AND MONEY AT A PREMIUM?
YOU NEED **DATA PROCESSING DIGEST**

Continuous search through computer, management, and trade magazines, reports, information services. Readable digests of the best material we find. Book reviews, monthly and yearly index, calendar, complete reference to sources. 12 monthly issues, $51.

Data Processing Digest
**Published Each Month Since 1955**

**Write for Information**

| |
|---|
| Name |
| Dept. |
| Company |
| Address |
| City State Zip |

Data Processing Digest, Inc.
6820 LA TIJERA BOULEVARD, LOS ANGELES, CALIFORNIA 90045 / PHONE (213) 776-4334

Let me depart from the format developed in this column over the past year to discuss in some detail the PLATO system. Recently I have had the opportunity to become very familiar with PLATO and its accompanying language, TUTOR, and would like to share some of the capabilities of the system with you. PLATO was developed at the University of Illinois, starting in 1960. It has been used extensively in instruction over the past few years, and the university has courseware (lesson material) written by several hundred faculty members. Several thousand hours of courseware is available to the students on a monthly basis.

PLATO is a computer-based education (CBE) system, which means it is CAI with a very high degree of sophistication. The term CBE implies more than just an assist to instruction; rather it suggests the actual instruction is accomplished by the machine. Assisting with course instruction is, of course, part of PLATO's abilities, but it is capable of managing instruction, and teaching by means of a very unusual terminal and a single language, TUTOR.

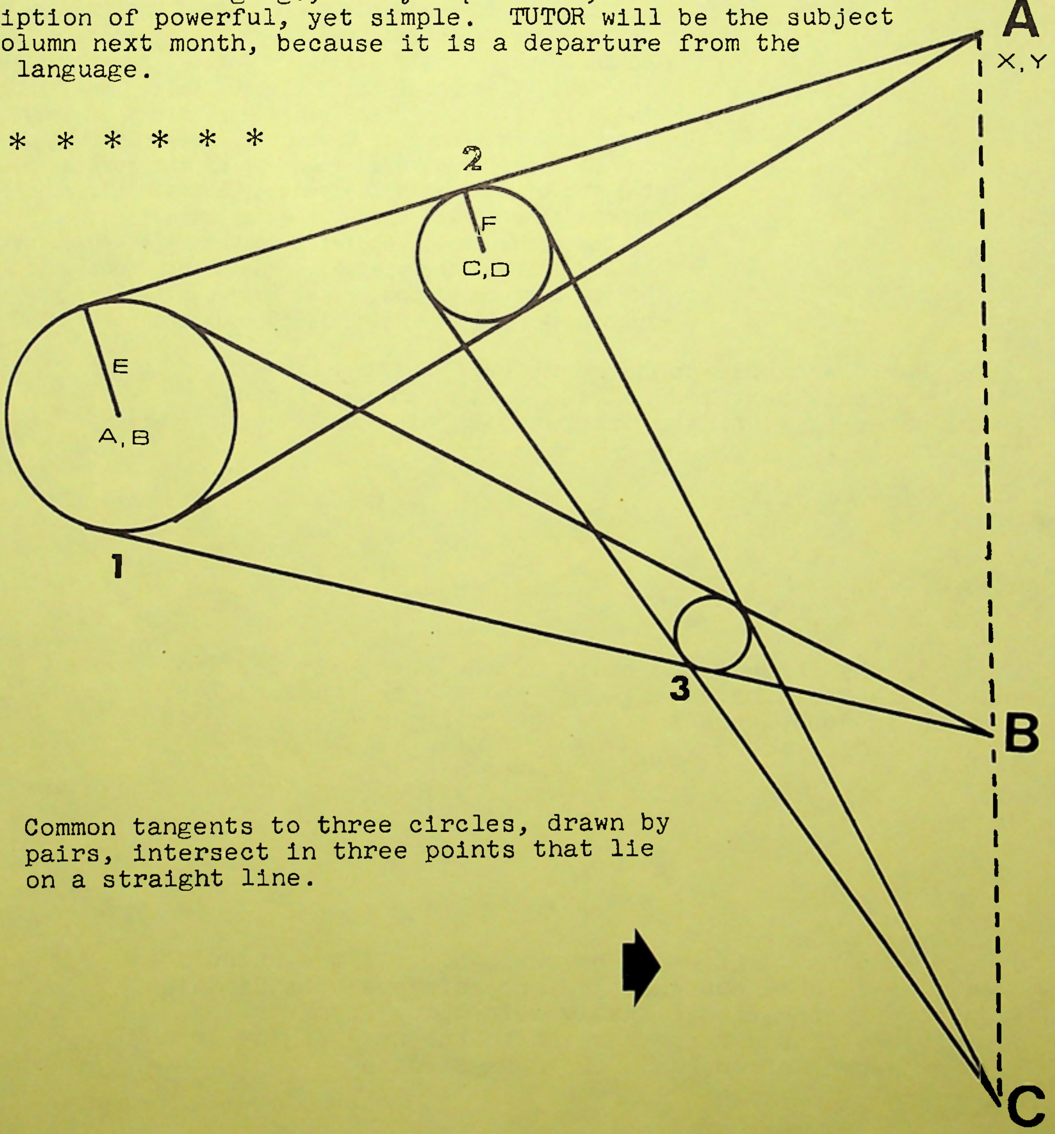
The PLATO terminal is unusual because it has a screen which is a plasma panel, together with a slide projector, an audio device, and a touch panel. The plasma panel consists of two plates of glass, each 8 1/2 inches square. On one is strung 512 vertical wires; on the other 512 horizontal wires. Between the two plates is sealed neon gas (plasma) and when a current strikes two wires at once, a spark jumps, exciting the neon, and a dot appears. The dot will remain until the current is stopped. Unlike most displays, therefore, the plasma panel does not need to be refreshed by the computer. The neon gives the characters an orange tint, which is unique in itself. The 512 x 512 dot grid allows for curves which are nearly continuous, as well as characters which have a high degree of resolution. The characters are formed on an 8 x 16 dot grid, for a total of 32, 64-character lines on the screen.

The keyboard allows for upper and lower case characters, and has 12 function keys of high usage, as well as several specialized ones. Subscripts, superscripts, addition, subtraction, multiplication, and division each are indicated by depressing a single key. (Subscripts and superscripts are displayed below and above the line in standard notation.) The other function keys are labelled with much more meaningful terms than on most terminals. NEXT will take the lesson to the next frame of information. BACK will restore the screen to its previous contents. HELP will give prompting information, hints, and general help, while DATA will supply more values or other data as a help to the student. The keyboard is arranged conveniently and is detached from the screen for ease in adjusting its height, etc.

The terminal also has a touch panel with a 16 x 16 grid of points that cause an input signal when touched. This allows for quick input, and is excellent for use with youngsters and people unfamiliar with terminals. The slide projector uses microfiche and shines a colored slide on to the back side of the plasma panel. Machine output can, therefore, be superimposed on top of the picture. Each of up to 256 slides can be accessed in four milliseconds. The audio device consists of a direct access disk which can contain 17 minutes of recorded information. Any point on the disk can be accessed within 4 milliseconds. In a language exercise, for example, a student can not only see the new grammar word, but hear it pronounced.

Of course, the heart of any CAI-type system is the language which drives it. This is one of the main reasons for the slow development of CAI; namely, lack of a really useful, yet simple, language. The TUTOR language, in my experience, does indeed fit the description of powerful, yet simple. TUTOR will be the subject of this column next month, because it is a departure from the usual CAI language.

* * * * * * * * *



Common tangents to three circles, drawn by pairs, intersect in three points that lie on a straight line.

➡

# Intersecting Tangents

In the figure, the external tangents to three circles (numbered 1, 2, and 3) are drawn, meeting at points A, B, and C. It is not difficult to show that A, B, and C must lie on a straight line.

This is problem 62 in Ingenious Mathematical Problems and Methods by Louis A. Graham (Dover Publications, 1959) and is attributed to Prof. John Edson Sweet, who furnished the elegant proof.

"It is related by K. K. Knaell of the Federal Glass Co., Columbus, Ohio, that when this problem was originally presented to Prof. Sweet he paused a moment and said, 'Yes, that is perfectly self evident.' Astonished, his friend asked him to explain how a problem that stopped most advanced students could be called self evident. Prof. Sweet, in effect, replied, 'Instead of three circles in a plane, imagine three balls lying on a surface plate. Instead of drawing tangents, imagine a cone wrapped around each pair of balls. The apexes of the three cones will then lie on the surface plate. On the top of the balls lay another surface plate. It will rest on the three balls and will be necessarily tangent to each of the three cones, and will contain the apexes of the three cones. Thus, the apexes of the three cones will lie in both of the surface plates, hence they must lie in the intersection of the two plates, which is of course a straight line.'"

For the circle centered at (a,b) with radius e, and the circle centered at (c,d) with radius f, the coordinates (x,y) of the intersection of the tangents are given by:

$$x = \frac{e}{e - f}(c - a) + a$$

$$y = \frac{e}{e - f}(d - b) + b$$

Thus, if the centers and radii of the three circles are known, the coordinates of A, B, and C can be calculated. If this is done, then the area of triangle ABC, which is given by the formula

$$A = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

should be zero. It would be surprising if the calculated area were to come out exactly zero, since a considerable amount of arithmetic is involved in going from the definitions of the three circles to the area of the triangle. It can happen, of course. For the circles

(6,20) radius 6
(26,18) radius 3
(30,10) radius 1

the points of intersection are (46, 16), (34.8, 8) and (32, 6) and these points form a zero-area triangle. On the other hand, for the circles

(6, 8) radius 5
(8, 9) radius 4.95
(5, 4) radius 4.94

the points of intersection, when inserted into the area determinant, yield an area of .07, using 8-digit arithmetic.

As the sizes of the radii approach each other, the points of intersection are forced farther out in the number system. If scientific notation and floating arithmetic are used (wherein the arithmetic system is not uniformly dense) those points can be pushed out as far as we please. As a result, we can make the error in the triangle's area as large as we please. For the circles

(6,8) radius 5
(8,9) radius 4.9999
(5,4) radius 4.9998

the error in the area is 900 square units.

**Back issues are still available.**

| JAN | FEB | MAR | APR | MAY | JUN | JUL | AUG | SEP | OCT | NOV | DEC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Vol. 1 | 1973 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | [✓ 21] | Vol. 2 | 1974 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | Vol. 3 | 1975 |

PC21-16

# Cycle Lengths

PROBLEM **70**

The table shows the three low-order digits of the first 120 powers of 2. The unit's digit repeats on a cycle of 4; the two low-order digits repeat on a cycle of 20; the three low-order digits repeat on a cycle of 100; and each successive digit to the left increases the cycle length by a factor of five.

What is the corresponding rule for powers of K, where K is 3, 5, 7, 9, 11,...?

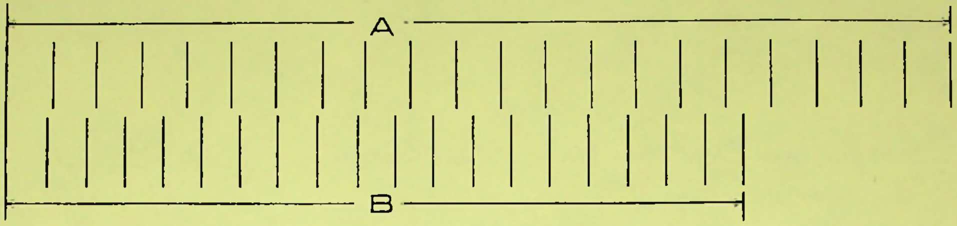
The problem as stated is intrinsically decimal. An easier problem is to determine the cycle length by bit positions for the binary version of powers of 3, 5, 7, 9, 11, and so on.

| | | | |
|---|---|---|---|
| 1 | 002 | 61 | 952 |
| 2 | 004 | 62 | 904 |
| 3 | 008 | 63 | 808 |
| 4 | 016 | 64 | 61[illegible] |
| 5 | 032 | 65 | 232 |
| 6 | 064 | 66 | 464 |
| 7 | 128 | 67 | 928 |
| 8 | 256 | 68 | 856 |
| 9 | 512 | 69 | 712 |
| 10 | 024 | 70 | 424 |
| 11 | 048 | 71 | 848 |
| 12 | 096 | 72 | 696 |
| 13 | 192 | 73 | 392 |
| 14 | 384 | 74 | 784 |
| 15 | 768 | 75 | 568 |
| 16 | 536 | 76 | 136 |
| 17 | 072 | 77 | 272 |
| 18 | 144 | 78 | 544 |
| 19 | 288 | 79 | 088 |
| 20 | 576 | 80 | 176 |
| 21 | 152 | 81 | 352 |
| 22 | 304 | 82 | 704 |
| 23 | 608 | 83 | 408 |
| 24 | 216 | 84 | 816 |
| 25 | 432 | 85 | 632 |
| 26 | 864 | 86 | 264 |
| 27 | 728 | 87 | 528 |
| 28 | 456 | 88 | 056 |
| 29 | 912 | 89 | 11[illegible] |
| 30 | 824 | 90 | 224 |
| 31 | 648 | 91 | 448 |
| 32 | 296 | 92 | 896 |
| 33 | 592 | 93 | 792 |
| 34 | 184 | 94 | 584 |
| 35 | 368 | 95 | 168 |
| 36 | 736 | 96 | 336 |
| 37 | 472 | 97 | 672 |
| 38 | 944 | 98 | 344 |
| 39 | 888 | 99 | 688 |
| 40 | 776 | 100 | 376 |
| 41 | 552 | 101 | 752 |
| 42 | 104 | 102 | 504 |
| 43 | 208 | 103 | 008 |
| 44 | 416 | 104 | 016 |
| 45 | 832 | 105 | 032 |
| 46 | 664 | 106 | 064 |
| 47 | 328 | 107 | 128 |
| 48 | 656 | 108 | 256 |
| 49 | 312 | 109 | 512 |
| 50 | 624 | 110 | 024 |
| 51 | 248 | 111 | 048 |
| 52 | 496 | 112 | 096 |
| 53 | 992 | 113 | 192 |
| 54 | 984 | 114 | 38[illegible] |
| 55 | 968 | 115 | 76[illegible] |
| 56 | 936 | 116 | 536 |
| 57 | 872 | 117 | 072 |
| 58 | 744 | 118 | 144 |
| 59 | 488 | 119 | 288 |
| 60 | 976 | 120 | 576 |

PROBLEM **71**

# Vernier

PC21-17



Length A (50 units) is divided into 21 equal parts by 20 ruled lines; these lines are red.

Length B (39 units) is divided into 19 equal parts by 18 ruled lines; these lines are blue. The red and blue lines are extended parallel to each other.

What is the distance between the closest pair of red and blue lines? (Answer: .082707 units.)

Call the parameter 21 C and the parameter 19 D. Write a subroutine whose arguments are A, B, C, and D, and whose output is the closest distance between the red and blue lines for those values.

| | |
|---|---|
| Log 21 | 1.322219294733919268007244161847751502683701260514661 |
| Ln 21 | 3.044522437723422996500597980365705434284575287404611 |
| $\sqrt{21}$ | 4.582575694955840006588047193728008488984456576767972 |
| $\sqrt[3]{21}$ | 2.758924176381120669465791108358521582252712086038936 |
| $\sqrt[5]{21}$ | 1.838416287252544036702798807184224042046754035404302 |
| $\sqrt[7]{21}$ | 1.544857660250173787974206529152693691396224973924725 |
| $\sqrt[10]{21}$ | 1.355882106693846755516285088817208916583633028863082 |
| $\sqrt[100]{21}$ | 1.030913419577617397615062351123987507595006895016803 |
| $e^{21}$ | 1318815734.483214697209998883745302785091444437380475 749065985845815069268613854287251343524759 |
| $\pi^{21}$ | 27551631842.873289070227728690708117606190899572552263 439098487520095958022825818623228571713 40 |
| $\tan^{-1}$ 21 | 1.523213223517913223429288975623265924632025705094221 |
| $21^{100}$ | 166697648439633735919597210805076652916730066782895 10 143313654693621330290703278666330330646324269063809 00 918045096212631206355582001 |

PC21-18

Printed by Argold Press, Inc.

*

## The Way To Learn Computing Is To Compute